Privacy-preserving recommendations for Online Social Networks using Trusted Execution Environments

Guilmour Rossi, Luiz Gomes-Jr[1], Marcelo Rosa[1], Keiko Fonseca [1]

`me@guilmour.org, gomesjr@dainf.ct.utfpr.edu.br, {mrosa,keiko}@utfpr.edu.br`

[1]Universidade Tecnológica Federal do Paraná
Curitiba – Brazil

Abstract. *Online Social Networks (OSN) have changed how individuals interact with each other and with organizations, offering means of communication, publication and consumption of information. As OSNs have become a substantial part of users' online activities, OSN providers have understood the value of the data being generated and exploited it to maximize profits. Recently, malicious agents have invested in the manipulation of OSN data to attain commercial advantages or influence public opinion with dangerous consequences. This paper describes our ongoing efforts towards the use of Trusted Execution Environments (TEE), more specifically Intel's SGX, for the management of recommendation engines for OSNs. Our solution focuses on protection of user data and prevention of misuse without compromising OSNs' functionality nor OSNs' revenue from advertisements. We describe the architecture of our system and report performance results that can be used to guide the selection of recommendation algorithms for execution under SGX.*

1. Introduction

Online Social Networks (OSN) have become commonplace in our technological landscape, being on dedicated services (e.g., Facebook, Linkedin) or in social features included in websites from most branches of industry and government (e.g., comments sections, shareable links). The interactions enabled by OSNs are clearly beneficial for users, providing entertainment, communication, self-expression (of opinions, interests, professional services, etc.), and discovery of products and services (based on the preferences of similar users).

However, by using these services, users leave behind a large trail of data that may be accessed and used in unethical practices [4]. Recent examples of such practices can be found in elections and referendums of major democracies, where malicious agents targeted citizens to spread misinformation and advance political interests related to ballot outcome. One way to prevent these problems is to encrypt all user data in such applications. This solution is, however, impractical because the business model of most OSNs is based on matching user preferences to advertisements. A solution to these problems must then combine increased privacy for users while still offering revenue opportunities for service providers.

In this paper, we describe an architecture for OSNs that uses Trusted Execution Environments, or TEEs, in order to shield user data from unauthorized access. In our solution, not even the service providers and advertisers can access the user data directly.

Instead, we offer content and advertisement recommendations as services in our Application Program Interface (API). In this scenario, service providers can still recommend content to users (e.g., movies for Netflix or posts for Facebook), and advertisers can match their ads to users' preferences.

The current implementations of TEEs have restrictions related to increased overhead and limited main memory available to the process. Therefore, it is important to assess the behavior of TEEs in the proposed scenario. In this paper, we present preliminary performance tests of various recommendation algorithms running on Intel SGX. The goal is to determine the algorithms that degrade the least under SGX and assess their applicability in the proposed platform.

The remainder of this paper is structured as follows: Section 2 describes related work and introduces important concepts. Section 3 provides details about the proposed architecture focusing on the aspects related to social recommendations. Section 4 shows experimental results for various recommendation algorithms running inside SGX enclaves. Finally, section 5 concludes the paper.

2. Background and related efforts

Governments around the world have grown concerned with the current state of privacy in OSNs. There has been many reports of loose privacy policies allowing misuse of user data with criminal or political purposes. The European Union has been the first to pass far-reaching legislation in an attempt to curb the abusive use of personal data. The General Data Protection Regulation (GDPR)[1] aims at protecting the privacy of individual users. While those are essential legal tools to guarantee privacy, our approach focuses on the technological aspects and aims at, by design, restricting access to user data.

On the technological front, there have been many efforts to improve privacy in OSNs. Many efforts focus on anonymization of user data [1], preserving a degree of privacy when publishing the social network data. These proposals aim at preventing abuse from third-party players but offer no protection against internal attacks or data breaches.

Other proposals focus on applying blockchains to store data and allow users to manage access rights to their private information. Zyskind et al. [10] developed a protocol that relies on a distributed blockchain to store and manage user data. While granting control of the data to the users, this type of approach still hands user data to providers, opening opportunities for abuse and misuse.

More recently, advances in hardware have allowed stronger privacy guarantees. Intel's SGX platform, for example, offers programmers with a secure environment where all computation is executed over encrypted memory. These environments, called enclaves, offer privacy even in worst-case scenarios where attackers have direct access to the provider's hardware. SGX [3] is currently the most comprehensive and commercially available implementation of a trusted execution environment (TEE). As a TEE, it has abilities of (i) containing a hardware encoded cryptographic key (known only by the module); (ii) remote attestation (which means two enclaves can trust each and exchange cryptographic keys), (iii) sealing data (which means once a data is sealed by the module, it can only

[1]https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en

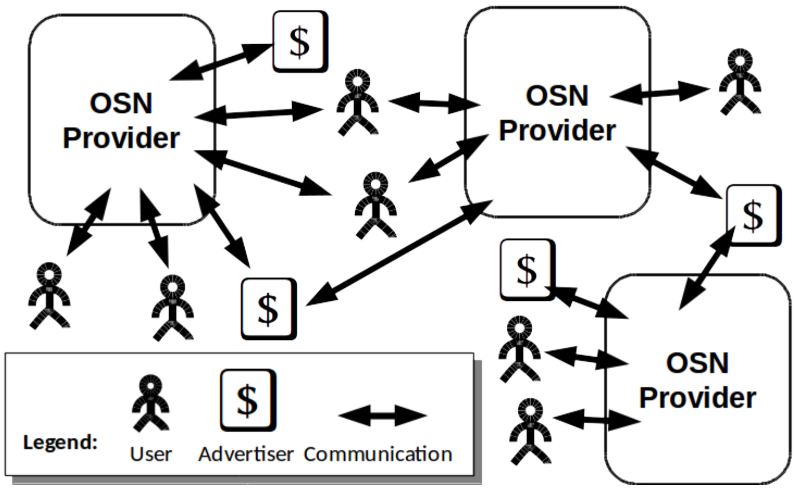


Figure 1. Overview of the interactions among the elements of the platform.

be unsealed by that module). To the developer's perspective, it ensures that nothing can tamper with the code running inside such modules.

There have been several proposals to harness these privacy guarantees for data processing platforms. OPAQUE [9] is a Spark based solution that employs SGX and obfuscation algorithms to guarantee privacy in a distributed processing context. The SecureCloud [8] project goes beyond the data processing scope offering a complete stack of tools that can be used to build SGX-based secure applications. The project offers application building blocks including programming language interpreters, communication protocols, and data management and processing services.

Neither OPAQUE nor SecureCloud intend to provide user-centered guarantees as those proposed here. The projects focus on protecting user data from attackers that may obtain physical access to provider's hardware. In their setting, there is no restriction to what providers or eventual partners can do with the data. Therefore, to properly tackle the current concerns regarding misuse of users data in OSN, it is necessary to design a more restrictive framework. In our proposal, even service providers do not have access to the social network data. To compensate for the added restrictions, the framework offers APIs for providers and advertisers to insert content into their OSNs. The framework itself is responsible for applying recommendation algorithms to suggest content and advertisement to users without the intervention of the provider.

3. Platform architecture

Our proposed architecture is based on three main roles: OSN providers, users and advertisers. OSN providers are organizations that offer services based on social network interactions. Examples of OSN providers include social networks such as Facebook and Twitter, but also encompass government and commerce websites that include social data for commentaries or recommendations. Users are persons with internet-enabled devices (smartphones, tablets, computers) that access the services of the OSN providers. Finally, advertisers are organizations that pay OSN providers to match their advertisement based on user preferences.

Figure 1 shows an example with interactions among these roles. As shown in the figure, users can access multiple OSN providers and, likewise, advertisers can promote content in multiple OSN providers. Details about internal elements are presented in the Figure 2. The main focus of this paper is the Social Security Platform, which is the component that processes user data and is described next. We also briefly describe the functionalities of other components in the next section.

3.1. Secure Social Platform (SSP)

The Secure Social Platform (SSP) is meant to be a generic platform that covers most application cases for online social networks, such as user management and content recommendation. Social application providers will be able to download and install the platform on their own servers and manage its performance as necessary. Being based on Trusted Execution Environments (TEE) technology, the platform is effectively a black box from the point of view of the providers. Providers will be able to configure the platform to suit their services, setting up content schemas and advertisement partners, and will be able to access anonymous statistics of visualization of content and ads. All the processing of data happens inside encrypted enclaves, protecting their privacy. The functionalities for user and content administration are exposed to providers as a service API.

Providers will also be able to build the frontend of their applications with any technology they want. However, the frontend server must provide XSLT (eXtensible Stylesheet Language Transformations) templates that will be used inside the Request Broker for user's data integration. Therefore, even though the provider is free to implement the user interface however it wants, as long as it provides adequate templates to be securelly merged with user data before delivery. At no point in this process the provider has access to user data.

Users and advertises will communicate with the Request Broker through an extension of the HTTP protocol that will encompass encryption, authentication and attestation procedures. The extension is called Social HTTPS (SHTTPS).

3.2. Privacy-preserving recommendations

The focus of this paper is in exploring one critical aspect of the Secure Social Platform: the recommendation of content to users. Recommendation algorithms are used for personalization, timeline construction, friendship suggestion, and advertisement placement. This type of algorithm is the most compute-intensive task in a OSN scenario and is central to effective user interaction and advertisement revenue.

To protect user data, we employ Intel's SGX architecture, currently the most advanced Trusted Execution Environments (TEEs) technology. SGX provides secure enclaves where software code runs in an encrypted region of the computer memory that is protected from attacks related to physical memory access. SGX, being a recent development, has limitations in terms of performance degradation and limited memory allocation. Therefore, it is important to test the limits of the technology in our intended scenario.

To simplify the deployment of our algorithms in SGX-enabled environments, we used SCONE [2]. SCONE is a secure container mechanism for Docker[2] that uses the SGX trusted execution to protect container processes from outside attacks. We implemented the

[2]https://www.docker.com/

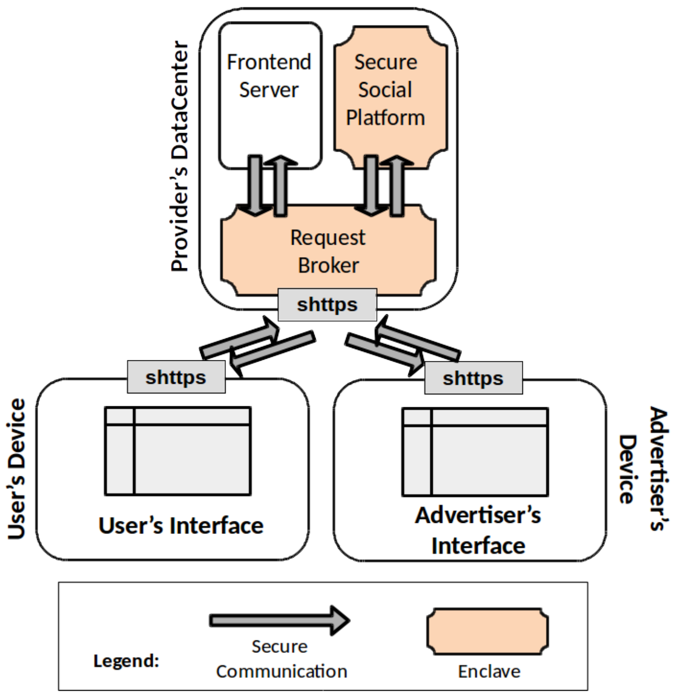


Figure 2. General architecture of the proposed platform.

service API in Python and employed recommendation algorithms from the Surprise [6] project.

4. Experimental procedure and results

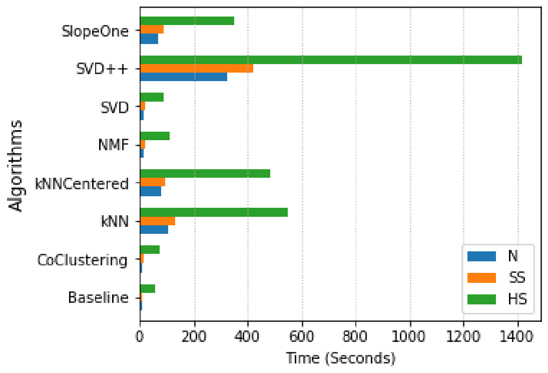
We study the performance of recommender algorithms in three distinct scenarios and using two different sizes of datasets. The three scenarios are meant to capture the performance overhead imposed by the use of SXG. The scenarios are:

1. Regular (N): running the code natively, using the standard Python interpreter.
2. Simulated SCONE (SS): interpreting the code inside the SCONE container, with the mode parameter set to simulated, where the execution proceeds with the SCONE functionalities but does not use the SGX hardware.
3. Hardware-based SCONE (HS): compilation is made in hardware mode, where the execution is forced by SCONE to run inside of the SGX enclave.

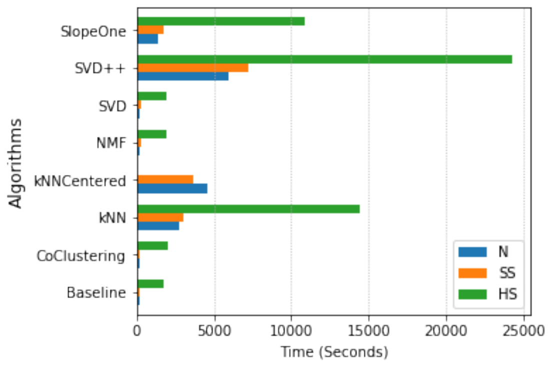
To build the prediction models, we used algorithms implemented in the Python Scikit-Surprise package [6], which offers a range of recommender algorithms, including variations of Singular-Value Decomposition (SVD) and k-Nearest Neighbor (kNN) approaches.

The two datasets used were obtained from the MovieLens datasets [5], one containing 100,000 ratings (1-5) from 943 users on 1682 movies and the other containing 1,000,209 ratings from 6,040 users on 3706 movies. We evaluated the time of basic execution of the API to train the data and build the predictions (in this case, the top 5 recommendations) for each user inside the database, including the time required by SCONE, when using it, to up the enclave and allocate memory.

Due to the limited size of the enclave page cache (EPC) [7], in this case 128 MB, we have to increase the SCONE's heap environment variable to 2 GB on the first dataset



(a) 100K Ratings dataset



(b) 1M Ratings dataset

Figure 3. Execution time of the algorithms on three distinct scenarios. Regular (N), with simulated SCONE (SS), and with hardware-enabled SCONE (HS).

and 16 GB on the second dataset. Is out of the scope of this paper considerations about accuracy and precision of the algorithms. Our goal is to examine performance degradation related to the use of SGX and its memory limitations.

All the experiments were performed on a GNU/Linux (kernel 4.15) based computer using an Intel Xeon E3-1280 v6 CPU with 4 cores at 3.90 GHz and 8 hyper-threads counting with 8 MB cache. The computer has 32 GB of RAM and a hard disk of 2 TB. The SCONE image used the operational system Alpine[3] version 3.6 and Python 3.5.4.

Table 1. Running times of the algorithms and proportional increase.

Algorithm	**Time (s)**					
	100K Dataset			**1M Dataset**		
	N	**SS**	**HS**	**N**	**SS**	**HS**
Baseline	8.3	12.8	59.6 (7.2*N; 4.7*SS)	142.2	180.4	1780 (12.6*N; 9.9*SS)
CoClustering	9.3	14.4	73.6 (7.9*N; 5.2*SS)	145.6	197.4	2000.7 (13.8*N; 10.2*SS)
kNN	107.8	135.1	548.9 (5.1*N; 4.1*SS)	2751.7	3025.8	14401.3 (5.3*N; 4.8*SS)
kNNCentered	81.1	98.5	484.3 (6*N; 5*SS)	4587.3	3619.0	82921,5 (18,1*N; 23*SS)
NMF	14.1	21.8	111.5 (7.9*N; 5.2*SS)	178.7	255.9	1925.3 (10.8*N; 7.6*SS)
SVD	14.7	22.3	92.5 (6.3*N; 4.2*SS)	194.6	263.2	1966.7 (10.2*N; 7.5*SS)
SVD++	324.0	422.6	1415.1 (4.4*N; 3.4*SS)	5966.8	7213.3	24277 (18*N; 3.4*SS)
SlopeOne	69.7	92.8	353.1 (5.1*N; 3.9*SS)	1354.1	1716.5	10877 (8.1*N; 6.4*SS)

Figure 3 shows execution time results for each algorithm. Baseline is a simple algorithm based on averages of user and item ratings and will not be discussed further. From the graphs, it can be seen that running the algorithms over SGX has a clear performance penalty. Also, kNN-based algorithms in general require a longer time to execute. The SVD-based algorithms (including NMF) performed well, except for SDV++ that requires a more complex data structure to represent ratings as well as more demanding calculations to derive the predictions. In the 1M tests, the kNNCentered algorithm finish after over 23 hours.

Table 1 shows the precise running times for the algorithms and also compares the proportional increase in time for the executions inside the SGX enclave (column HS, in

[3]https://alpinelinux.org/about

Table 2. Average RMSE and MAE of the algorithms.

Algorithm	**Dataset**			
	100K		1M	
	RMSE	MAE	RMSE	MAE
Baseline	0.944	0.748	0.909	0.719
CoClustering	0.963	0.753	0.915	0.717
kNN	0.980	0.774	0.923	0.727
kNNCentered	0.951	0.749	0.929	0.738
NMF	0.963	0.758	0.916	0.724
SVD	0.934	0.737	0.873	0.686
SVD++	0.920	0.722	0.862	0.673
SlopeOne	0.946	0.743	0.907	0.715

Source: Scikit-Surprise documentation [6].

parenthesis). Here it can be seen a clear distinction between the algorithms based on SVD and kNN: the time for SVD algorithms degrades more (when compared with non-SGX executions) as the dataset grows. For example, comparing with regular (N) execution, SVD took 6.3 times longer to execute in the 100K and 10.2 times longer in the 1M dataset. The KNN algorithm maintained the approximately 5 times penalty throughout the tests. The CoClustering algorithm was the fastest in both datasets but, similarly to SVD methods, it degrades more when using SGX. This shows the importance of benchmarking the algorithms in the SGX context, since standard benchmarks would not reveal this SGX-specific trend.

As a reference, we included in Table 2 recommendation quality assessments for each algorithm. SVD-based algorithms tend to provide more accurate predictions in terms of Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Therefore, this must be considered when choosing the algorithm for a task with high accuracy requirements.

5. Conclusion

This paper described our architecture for privacy-preserving Online Social Networks (OSNs), detailing the usage scenario, roles and components. The main focus of the paper is testing a critical component of OSNs: the recommendation algorithms used in several tasks such as content suggestion and ad placement. Since our platform takes advantage of Intel's SGX technology, it is important to assess the performance of recommendation algorithms in the secure environment.

Our experiments have shown that different classes of algorithms respond differently to the SGX environment. This is likely caused by SGX's enclave page cache (EPC) limitations. The tests suggest that SVD-based algorithms tent to degrade more, in relative terms, under SGX for larger datasets. However, in absolute terms, the pure SVD algorithm (and also the related NMF algorithm) was shown to be faster than kNN methods even in the larger dataset. The tests are inconclusive in whether the stronger degradation under SGX would favor kNN methods for even larger datasets (millions of users and items). To perform this type of test, we will be implementing a parallel version of the architecture and employing parallel variations of the algorithms.

Other future efforts will focus on refining the API to be applied in real world OSNs: including authentication steps, error handling, and secure secondary memory storage.

The tests presented here show the practicability of our approach. We consider the performance penalty for the best performing algorithms to be reasonable given the improved security guarantees under SGX. We expect that this type of approach will play an important role in protecting user data in OSNs.

6. Acknowledgments

This research is being performed in the context of the SecureCloud project. The SecureCloud project has received funding from the European Union's Horizon 2020 research and innovation program and was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under grant agreement number 690111. This work was partially funded by the EU-BR SecureCloud project (MCTI/RNP 3rd Coordinated Call).

Referências

[1] J. H. Abawajy, M. I. H. Ninggal, and T. Herawan. Privacy preserving social network data publication. *IEEE Communications Surveys and Tutorials*, 18(3):1974–1997, 2016.

[2] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. Stillwell, D. Goltzsche, D. M. Eyers, R. Kapitza, P. R. Pietzuch, and C. Fetzer. Scone: Secure linux containers with intel sgx. In *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016.

[3] V. Costan and S. Devadas. Intel sgx explained. Technical Report 2016/086, Cryptology ePrint Archive, 2016.

[4] M. Fire, R. Goldschmidt, and Y. Elovici. Online social networks: Threats and solutions. *IEEE Communications Surveys and Tutorials*, 16(4):2019–2036, 2014.

[5] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, Dec. 2015.

[6] N. Hug. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

[7] I. R. Intel. Software guard extensions sdk for linux* os, revision 1.5.

[8] F. Kelbert, F. Gregor, R. Pires, S. Köpsell, M. Pasin, A. Havet, V. Schiavoni, P. Felber, C. Fetzer, and P. R. Pietzuch. Securecloud: Secure big data processing in untrusted clouds. In D. Atienza and G. D. Natale, editors, *DATE*, pages 282–285. IEEE, 2017.

[9] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *14th USENIX Symposium on Networked Systems Design and Implementation*, pages 283–298, 2017.

[10] G. Zyskind, O. Nathan, and A. Pentland. Decentralizing privacy: Using blockchain to protect personal data. In *IEEE Symposium on Security and Privacy Workshops*, pages 180–184. IEEE Computer Society, 2015.